Using Transfer Metadata to Create a Memento

Posted At : June 17, 2008 7:05 AM | Posted By : Bob Silverberg
Related Categories: ColdFusion, Transfer

This was asked today on the **Transfer mailing list**:

```
Looking through the documentation I couldn't see any generated methods which will return all the objects properties as a structure of 'propertyname:value'.


Does a method get generated which handles this? If not, what do you see as the best way to build one?
```

I thought that my response was worthy of a blog post, so here goes.

Firstly, there is a getMemento() method that gets generated for all Transfer objects, which returns this type of struct. The problem is that it is undocumented and therefore unsupported. This method could change and/or disappear at any time, so it's not really safe to use. I use it occasionally while developing and debugging, but I'd never use it in real application code.

Transfer does provide access to its own internal metadata via **getTransferMetaData()**, which you can call on the base Transfer class. This metadata is chock full of useful information which I've made use of in a number of situations. I too needed to generate a simple struct that contained property name/value pairs, so I wrote a method which I've placed in my AbstractTransferDecorator which serves that purpose. Here's the code, broken up into sections to allow for some explanation:

```
<cffunction name="copyToStruct" access="public" output="false" returntype="void" hint="Copies data from the TO into a Struct">

 <cfargument name="theStruct" type="any" required="false" default="#StructNew()#" hint="A struct to receive the data" />

 <cfargument name="Overwrite" type="any" required="false" default="true" hint="Should existing values in the struct be overwritten?" />


 <cfset var TransferMetadata = getTransfer().getTransferMetaData(getClassName()) />

 <cfset var Properties = TransferMetadata.getPropertyIterator() />

 <cfset var PrimaryKey = TransferMetadata.getPrimaryKey() />

 <cfset var theProperty = 0 />

 <cfset var varName = 0 />

 <cfset var varType = 0 />

 <cfset var varValue = 0 />

 <cfset var i = 0 />
```

So first I grab the Transfer metadata for this class. getClassName() is a **generated method** that exists for every Transfer object. I know I want to populate keys in my structure for all of the properties of my Transfer object, so I grab the PropertyIterator from the metadata. My primary key, identified as an id in transfer.xml will not be included in the PropertyIterator, so I need to ask for that separately.

```
<cfset varName = PrimaryKey.getName() />

 <cfinvoke component="#this#" method="get#varName#" returnvariable="varValue" />

 <cftry>

  <cfset StructInsert(arguments.theStruct,varName,varValue,arguments.Overwrite) />

  <cfcatch type="any"></cfcatch>

 </cftry>
```

I use the PrimaryKey from the metadata to determine the name of id, which I then use to call the associated get() method. I then add this name/value pair to the struct. Because I want this routine to be able to add information to an existing struct, I may not always want it to overwrite any existing keys in the struct, which is the reason that I'm allowing the caller to pass in an argument for overwrite. The StructInsert function allows you to specify overwrite as either true or false, so this allows me to implement this logic. Unfortunately, if you specify overwrite as false and the key actually exists an exception is thrown, which is the reason that I've wrapped my call to StructInsert in a cftry. There's probably a more elegant way of doing this, but I haven't taken the time to find one... yet.

```
<cfloop condition="#Properties.hasnext()#">

  <cfset theProperty = Properties.next() />

  <cfset varName = theProperty.getName() />

  <cfinvoke component="#this#" method="get#varName#" returnvariable="varValue" />

  <cftry>

   <cfset StructInsert(arguments.theStruct,varName,varValue,arguments.Overwrite) />

   <cfcatch type="any"></cfcatch>

  </cftry>

 </cfloop>
```

I loop through the PropertyIterator, doing the same thing as above - getting the value with a getter and then adding it to the struct.

At this point I have my id and all properties copied into my struct. An issue I encoutered is that I have some custom methods in my decorator and they are not exposed as properties to Transfer. For example, you might define a getAge() method which calculates a person's age based on their date of birth and the current date. if you want that copied to the struct as well, it won't happen if you rely solely on Transfer metadata. To allow for this I define a variable in the Configure() method of my object's decorator called ExtraProperties, and I populate it with an array of property names. These names correspond with the custom getters that I created, and for which I want values copied into my struct.

```
<cfif IsDefined("variables.myInstance.ExtraProperties") AND IsArray(variables.myInstance.ExtraProperties) AND ArrayLen(variables.myInstance.ExtraProperties)>
```

```
  <cfloop from="1" to="#ArrayLen(variables.myInstance.ExtraProperties)#" index="i">

   <cfinvoke component="#this#" method="get#variables.myInstance.ExtraProperties[i]#" returnvariable="varValue" />

   <cftry>

    <cfset StructInsert(arguments.theStruct,variables.myInstance.ExtraProperties[i],varValue,arguments.Overwrite) />

    <cfcatch type="any"></cfcatch>

   </cftry>

  </cfloop>

 </cfif>
```

And that's the end of the method.

```
</cffunction>
```

For anyone interested in cutting and pasting, here's the whole function, with a few extra comments:

```
<cffunction name="copyToStruct" access="public" output="false" returntype="void" hint="Copies data from the TO into a Struct">

 <cfargument name="theStruct" type="any" required="false" default="#StructNew()#" hint="A struct to receive the data" />

 <cfargument name="Overwrite" type="any" required="false" default="true" hint="Should existing values in the struct be overwritten?" />


 <!--- Get the MetaData and Properties --->

 <cfset var TransferMetadata = getTransfer().getTransferMetaData(getClassName()) />

 <cfset var Properties = TransferMetadata.getPropertyIterator() />

 <cfset var PrimaryKey = TransferMetadata.getPrimaryKey() />

 <cfset var theProperty = 0 />

 <cfset var varName = 0 />

 <cfset var varType = 0 />

 <cfset var varValue = 0 />

 <cfset var i = 0 />


 <!--- Put the Id into the Struct --->

 <cfset varName = PrimaryKey.getName() />

 <cfinvoke component="#this#" method="get#varName#" returnvariable="varValue" />

 <!--- Need cftry because if overwrite is false an error can be thrown --->

 <cftry>

  <cfset StructInsert(arguments.theStruct,varName,varValue,arguments.Overwrite) />

  <cfcatch type="any"></cfcatch>

 </cftry>

 <!--- Put the properties into the theStruct --->

 <cfloop condition="#Properties.hasnext()#">

  <cfset theProperty = Properties.next() />

  <cfset varName = theProperty.getName() />

  <cfinvoke component="#this#" method="get#varName#" returnvariable="varValue" />

  <cftry>

   <cfset StructInsert(arguments.theStruct,varName,varValue,arguments.Overwrite) />

   <cfcatch type="any"></cfcatch>

  </cftry>

 </cfloop>

 <!--- Add the extra properties into the Struct --->

 <cfif IsDefined("variables.myInstance.ExtraProperties") AND IsArray(variables.myInstance.ExtraProperties) AND ArrayLen(variables.myInstance.ExtraProperties)>

  <cfloop from="1" to="#ArrayLen(variables.myInstance.ExtraProperties)#" index="i">

   <cfinvoke component="#this#" method="get#variables.myInstance.ExtraProperties[i]#" returnvariable="varValue" />

   <cftry>

    <cfset StructInsert(arguments.theStruct,variables.myInstance.ExtraProperties[i],varValue,arguments.Overwrite) />

    <cfcatch type="any"></cfcatch>

   </cftry>

  </cfloop>

 </cfif>

</cffunction>
```